Computers & Operations Research 38 (2011) 1543–1555

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

The open capacitated arc routing problem

Fábio Luiz Usberti [a,*], Paulo Morelato França [b], André Luiz Morelato França [a]

[a] School of Electrical and Computer Engineering, Campinas State University, 13083-852 Campinas, SP, Brazil
[b] Department of Mathematics, Statistics and Computing, São Paulo State University, 19060-900 Presidente Prudente, SP, Brazil

ARTICLE INFO

Available online 20 January 2011

Keywords:
Arc routing
Computational complexity
Path-scanning heuristic

ABSTRACT

The Open Capacitated Arc Routing Problem (OCARP) is a NP-hard combinatorial optimization problem where, given an undirected graph, the objective is to find a minimum cost set of tours that services a subset of edges with positive demand under capacity constraints. This problem is related to the Capacitated Arc Routing Problem (CARP) but differs from it since OCARP does not consider a depot, and tours are not constrained to form cycles. Applications to OCARP from literature are discussed. A new integer linear programming formulation is given, followed by some properties of the problem. A reactive path-scanning heuristic, guided by a cost-demand edge-selection and ellipse rules, is proposed and compared with other successful CARP path-scanning heuristics from literature. Computational tests were conducted using a set of 411 instances, divided into three classes according to the tightness of the number of vehicles available; results reveal the first lower and upper bounds, allowing to prove optimality for 133 instances.

1. Introduction

The Capacitated Arc Routing Problem (CARP), proposed by Golden and Wong [1], is a combinatorial optimization problem defined in a connected undirected graph $G(V,E)$ with non-negative costs and demands on edges. There is a fleet of identical vehicles with limited capacity that must service all edges with positive demand (required edges). The objective is to search for a set of minimum cost tours that begin and must return to a distinguished node, called depot.

The CARP belongs to the class of NP-hard problems, and it has been shown that even the $\frac{3}{2}-$approximation for the CARP is also NP-hard [1]. Some of the heuristics which perform well in most cases are path-scanning [2,3], augment-merge [1], and augment-insert [4]. Even better solutions were obtained through metaheuristics such as tabu search [5–7], genetic algorithm [8], hybrid tabu-scatter search [9], guided local search [10], and a variable neighborhood descent algorithm [11]. In terms of approximate algorithms, the current best approximation factor is $\frac{7}{2}-\frac{3}{D}$ [12], where D is the vehicle capacity. There is an exact algorithm based upon a branch-and-bound paradigm [13], and another which transforms the CARP into the capacitated vehicle routing problem (CVRP), and solve it using a branch-and-cut-and-price algorithm [14]. These exact approaches, however, can only solve relatively small size instances. Furthermore, there are algorithms specialized in determining lower bounds for the CARP [15–21]. An overview on the CARP complexity, polyhedral results, exact, approximate, and heuristic algorithms can be found in [22–26].

As Dror [23] observed, there are two CARP versions with respect to the number of vehicles. In the first one, which corresponds with the original CARP conception, the number of vehicles is a fixed parameter. The second version considers the number of vehicles as a decision variable which means that, in this version, CARP algorithms account of an unlimited fleet of vehicles. Welz [27] observed that determining whether a feasible solution exists for a given number of tours is already NP-hard, since it requires solving a bin-packing problem. This may be the reason why many state-of-the-art heuristics deal with the second CARP version.

This work introduces the Open Capacitated Arc Routing Problem (OCARP), similar to CARP but different because tours are not constrained to form cycles, and therefore both open and closed tours are permitted. In both problems there are required edges, which must be serviced, and non-required edges, only used when necessary, i.e., on the path from one required edge to another. Considering that a tour start at node v_s and terminate at node v_t, in CARP $v_s = v_t = v_0$ for all tours, hence a particular case of OCARP. Consequently, the OCARP can be seen as a CARP generalization, and in the best of our knowledge this problem has never been formally reported in literature, in spite of important practical problems could be easily modeled as an OCARP.

In Section 2 it is given the OCARP formal description followed by some applications in Section 3. An integer linear programming (ILP) is formulated in Section 4, deriving also some interesting properties. The OCARP complexity is analyzed in Section 5 and

* Corresponding author. Tel.: +55 19 35213881.
E-mail address: fusberti@gmail.com (F.L. Usberti).

0305-0548
doi:10.1016/j.cor.2011.01.012

a solving strategy given in Section 6. Path-scanning heuristics for the CARP are reviewed in Section 7, and a reactive version for the OCARP is proposed. Computational experiments were conducted and the results presented in Section 8. Conclusions and future works close this article in Section 9.

2. Problem statement

Let $G(V,E)$ be an undirected connected graph where non-negative costs c_{ij} and demands d_{ij} are assigned to each edge $e = [v_i, v_j]$. All edges with positive demands, called required edges ($R \subseteq E$), must be serviced once by a single vehicle. A fleet of M identical vehicles with limited capacity D is available. While traversing the graph, a vehicle might (i) service an edge, which deducts the demand from the vehicle capacity and increases the solution cost or (ii) deadhead an edge, which only increases the solution cost. A vehicle tour is defined by a set of directed edges (arcs) traversed by that vehicle, and here both open and closed tours are considered, i.e., an OCARP tour may start and end at distinct nodes.

A feasible OCARP solution is thus formed by a family of tours, which services all required edges and does not violate any vehicle capacity (Fig. 1). The objective of OCARP is to find the minimum cost family of tours.

3. Applications

Many combinatorial optimization problems can be represented as an OCARP instance. For the sake of space limitations this section selects only two problems from literature which revealed themselves as interesting applications for the OCARP.

3.1. Meter reader routing problem

The Meter Reader Routing Problem (MRRP) interests major electric, water and gas distribution companies which periodically needs to meter read their clients. Like the OCARP, the MRRP does not consider a depot since the employees responsible for metering, called meter readers, are taken by auto from the office to the address of their first card, and after completing their routes, they take public transportation to return home. The objective is to find a set of tours for meter readers, with limited amount of working time, that visit every street segment containing clients in a minimum traversal time. Service time is incurred whenever an employee meter reads, while a shorter deadheading time is computed when the employee is not reading. All street segments have positive deadhead time, however, some may have zero service time, which means that there are no clients on that segment.

Stern and Dror [28] routed meter readers for the state power company from Beersheva, Israel, and developed a *route first, cluster second* heuristic, where initially the problem is treated as non-capacitated, and a single route covers all required edges. This single route is partitioned into segments, each designated to a meter reader.

Wunderlich et al. [29] routed meter readers for the Southern California Gas Company (SOCAL) from Los Angeles, USA, using an adapted arc partitioning algorithm developed and further improved by Bodin and Levy [30,31]. In their algorithm the graph is partitioned into meter readers territories, followed by tour construction inside each territory. This algorithm represents a reverse strategy compared to Stern and Dror proposition, i.e., a *cluster first, route second* heuristic.

The transformation from MRRP to OCARP comes naturally:

1. Vehicles correspond to meter readers, with capacity equal to their working time.
2. Vertices correspond to street intersections.
3. Edges correspond to street segments.
4. Edge cost represents deadheading time.
5. Edge demand represents servicing time.

A singularity of the MRRP is that even when the reader is not servicing an edge, he is using part of his working time by deadheading. Therefore the corresponding OCARP vehicle should have its remaining capacity decreased not only when servicing, but also when deadheading.

3.2. Cutting path determination problem

In the Cutting Path Determination Problem (CPDP) the trajectories of a set of blowtorches must be determined for a cut pattern on a quadrilateral steel plate in order to produce a predefined set of polygonal pieces in minimum time. A piece is produced when its shape is fully traversed by one or more blowtorches. These blowtorches have a limited amount of energy to spend and must not traverse the interior of any shape, but they may dislocate above the plate level, reflecting additional elevating and lowering maneuvers times. Moreira et al. [32] investigated a version of CPDP using the concept of a dynamic rural postman problem. In this dynamic version, the related graph changes during the cutting process because when a piece is produced it falls off into a special container, therefore giving new possible paths for the blowtorches to take.

The CPDP may also be modeled as an OCARP through the following transformation:

1. Graph vertices correspond to polygons vertices.
2. Graph edges correspond to polygons edges.
3. Non-required edges are formed by the set of rectilinear trajectories between all pairs of vertices. This set of non-required edges is partitioned into *upper* and *lower* non-required edges. The lower edges are those which do not overrun the interior of any polygon.

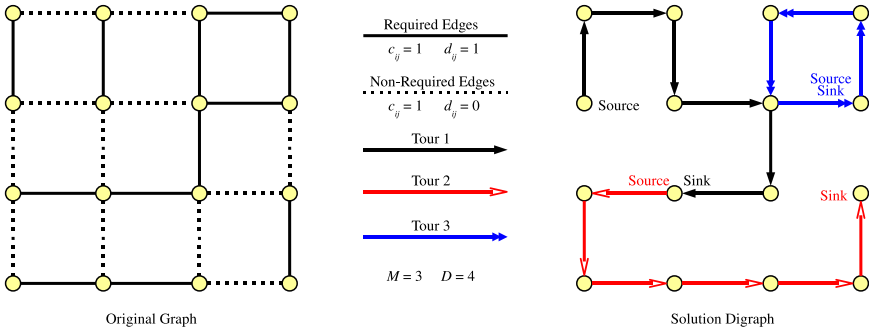

Fig. 1. An OCARP instance and a corresponding feasible solution.

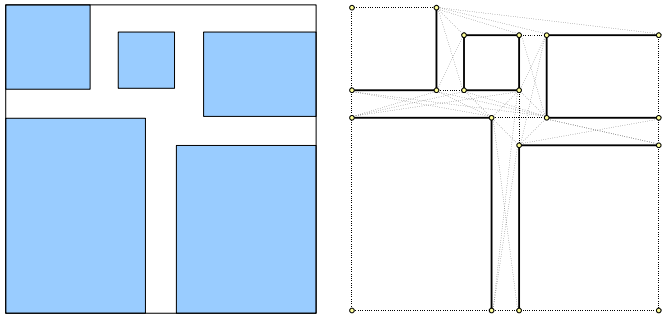

Fig. 2. Transforming a CPDP instance into an OCARP instance.

The upper edges represent a blowtorch dislocation above the plate level.
4. Edge cost corresponds to traversal time, reminding that non-required upper edges imply additional times due to elevating and lowering maneuvers.
5. Demand represents the energy spent to cut through the plate (while traversing required and lower non-required edges).
6. Vehicles correspond to blowtorches, with capacity equivalent to the amount of energy they are allowed to spend.
7. The objective function seeks the minimum makespan.

This transformation has polynomial complexity bounded by $O(n^2)$, where n is the number of vertices. Fig. 2 shows an example of a cutting pattern (plate with the shapes to be produced) and the equivalent OCARP graph. Required edges are represented by continuous lines and non-required by dotted lines. Given the large amount of upper non-required edges, these were omitted.

4. ILP model

In this section an integer linear programming model for OCARP is proposed, starting with the following CARP model from Golden and Wong [1] and then discussing the differences. This model considers only directed variables, hence each edge $(i,j) \in E$ from the undirected CARP graph $G(V,E)$ is treated as two arcs (i,j) and (j,i). It is assumed that node v_0 is the depot, and that the number of vehicles M is a fixed parameter. $N(i)$ denotes the nodes adjacent to node i in G.

Two sets of decision variables are defined: $x_{ij}^k = 1$ if tour k traverses arc (i,j), $x_{ij}^k = 0$ otherwise; $l_{ij}^k = 1$ if tour k services arc (i,j), $l_{ij}^k = 0$ otherwise. There are the auxiliary variables u_S^k and v_S^k ($S \subseteq V$, $\tilde{S} = V \backslash S$): if $u_S^k = 0$ then tour k does not have a sufficient number of arcs to form a cycle in S; if $v_S^k = 0$ then tour k traverses the cut-set $(S,\tilde{S})$. It is worth mentioning that the converse for these two conditional statements does not hold true, meaning that it is not possible to predict the values of u_S^k and v_S^k given tour k arcs in S and $(S,\tilde{S})$.

(CARP)

$$\min \quad \sum_{k=1}^{M} \sum_{(i,j) \in E} c_{ij} x_{ij}^k \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in N(i)} (x_{ji}^k - x_{ij}^k) = 0 \quad (i \in V, k \in \{1,\ldots,M\}) \tag{2}$$

$$x_{ij}^k \geq l_{ij}^k \quad ((i,j) \in R, k \in \{1,\ldots,M\}) \tag{3}$$

$$\sum_{k=1}^{M} (l_{ij}^k + l_{ji}^k) = 1 \quad ((i,j) \in R) \tag{4}$$

$$\sum_{(i,j) \in R} d_{ij} l_{ij}^k \leq D \quad (k \in \{1,\ldots,M\}) \tag{5}$$

$$\left.\begin{array}{l} \sum_{(i,j) \in (S,S)} x_{ij}^k - |S|^2 u_S^k \leq |S| - 1 \\ \sum_{(i,j) \in (S,\tilde{S})} x_{ij}^k + v_S^k \geq 1 \\ u_S^k + v_S^k \leq 1 \end{array}\right\} \quad (S \subseteq V \backslash \{v_0\}, \tilde{S} = V \backslash S, k \in \{1,\ldots,M\}) \tag{6}$$

$$x_{ij}^k \in \{0,1\} \quad ((i,j) \in E, k \in \{1,\ldots,M\}) \tag{7}$$

$$l_{ij}^k \in \{0,1\} \quad ((i,j) \in R, k \in \{1,\ldots,M\}) \tag{8}$$

$$u_S^k, v_S^k \in \{0,1\} \quad (k \in \{1,\ldots,M\}, S \subseteq V \backslash \{v_0\}) \tag{9}$$

The objective function (1) minimizes the solution total cost. Constraints (2) maintain routes continuity (every node must have equal indegree and outdegree). Constraints (3) state that serviced arcs must also be traversed; (4) force each required edge (represented by two arcs) to be serviced in a unique direction and by a single vehicle; (5) are the capacity constraints; and (6) eliminate illegal subcycles. For a CARP illegal subcycle to occur, referring to some tour k, two necessary conditions must be satisfied under subset S, strictly containing the subcycle nodes (except the depot):

1. The number of tour arcs in S is at least $|S|$, otherwise there would not be enough arcs to form the subcycle:
$$\sum_{(i,j) \in (S,S)} x_{ij}^k \geq |S| \tag{10}$$
2. There are no arcs in the cut $(S,\tilde{S})$, meaning that the subcycle is disconnected from the tour:
$$\sum_{(i,j) \in (S,\tilde{S})} x_{ij}^k = 0 \tag{11}$$

Constraints (6), by using the auxiliary variables u_S^k and v_S^k, state that at most one of the two conditions (10) or (11) can be satisfied for any $S \subseteq V \backslash \{v_0\}$.

Constraints (2) are not valid for OCARP since open tours are feasible. Each directed open tour contains a source and a sink nodes, which can be detected by the difference between their indegree and outdegree (if the tour is a cycle, then any node can be the source and the sink). That said, valid continuity constraints for the OCARP are represented by (12)–(14).

$$\sum_{j \in N(i)} (x_{ij}^k - x_{ji}^k) \leq \alpha_i^k \quad (i \in V, k \in \{1,\ldots,M\}) \tag{12}$$

$$\sum_{i \in V} \alpha_i^k \leq 1 \quad (k \in \{1,\ldots,M\}) \tag{13}$$

$$\alpha_i^k \in \{0,1\} \quad (i \in V, k \in \{1,\ldots,M\}) \tag{14}$$

The auxiliary variable $\alpha_i^k = 1$ if node i from tour k is the source, $\alpha_i^k = 0$ otherwise. While constraints (12) detect which nodes are sources, constraints (13) declare that each tour may contain at most one source. It should be noticed that there is no need to restrain the sink nodes, once they are implicitly restricted by the graph degree balance.

Constraints (6) are based on the fact that a cycle is illegal for CARP if it is disconnected from every tour. This is not valid for OCARP, since such a disconnected cycle can be legal, as long as it represents a whole tour (see Fig. 3). Surely, a disconnected cycle in any subset $S \subseteq V$ is valid for OCARP if S contains the tour source. Therefore, it is possible to extend CARP illegal subcycle necessary

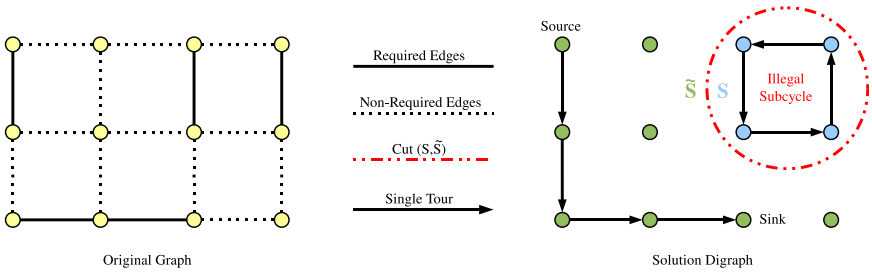

Fig. 3. OCARP illegal subcycle.

conditions (10) and (11) to OCARP, under subset S strictly containing the subcycle nodes, by comprising a third condition (15).

3. There is no source node in S.

$$\sum_{i \in S} \alpha_i^k = 0 \qquad (15)$$

With this third condition, an OCARP adaptation of the subcycle elimination constraints is given in (16) and (17):

$$\left.\begin{array}{l} \sum_{(i,j) \in (S,S)} x_{ij}^k - |S|^2 u_S^k \le |S| - 1 \\ \sum_{(i,j) \in (S,\tilde{S})} x_{ij}^k + v_S^k \ge 1 \\ \sum_{i \in S} \alpha_i^k + w_S^k \ge 1 \\ u_S^k + v_S^k + w_S^k \le 2 \end{array}\right\} \quad (S \subseteq V, \tilde{S} = V \backslash S, k \in \{1, \ldots, M\}) \qquad (16)$$

$$u_S^k, v_S^k, w_S^k \in \{0,1\} \quad (k \in \{1, \ldots, M\}, S \subseteq V \backslash \{1\}) \qquad (17)$$

Replacing constraints (2), (6), and (9) by (12)–(14), (16), and (17) in CARP model, leads to a valid OCARP integer linear programming model.

The binary constraints (7) under the decision variables x_{ij}^k may induce that an OCARP tour could traverse an arc only once, which is not true. However, that assumption is valid when considering an optimal solution.

Some interesting OCARP properties are:

Property 1. *Given an OCARP instance with M vehicles and at least M required edges, there exists an optimal solution which uses all vehicles.*

Proof. Let there be an optimal solution which uses less than M vehicles. Since there are more than $(M-1)$ required edges, at least one vehicle is traversing two or more required edges. In this case, we can split this vehicle tour in two, leaving at least one required edge in each tour, and the solution cost would remain the same. This procedure can be repeated until all vehicles attend a required edge, giving the property correctness. □

A consequence of this property is that instances where $M \ge |R|$ are trivially solvable by assigning one vehicle per required edge. Therefore, unlike CARP which admits M being a decision variable, OCARP has only meaning if M is a fixed parameter.

Property 2. *Given an OCARP instance, there exists an optimal solution in which all tours start and finish with required edges.*

Proof. Consider an optimal tour with a terminal non-required edge. If this edge is simply removed from the tour, feasibility is maintained. This process can be iterated until there are no longer any terminal non-required edges, giving the property correctness. □

A consequence of this property is that if there is an optimal cyclic tour, then all of its edges are required (excluding non-required edges with zero cost).

5. Complexity study

Through a polynomial reduction CARP $\le_p$ OCARP, this section intends to prove that the latter is at least as hard as the former. Furthermore, knowing if a problem, such as OCARP, is NP-hard, justifies the employment of (meta)heuristics to find good quality solutions, since exact methods are likely inadequate to solve realistic sized instances.

Theorem 1. *The CARP can be reduced polynomially into OCARP.*

Starting from any CARP instance $G(V,E)$, with M vehicles and a depot node v_0, add $2M$ dummy nodes (V_0) and $2M$ dummy required edges (R_0), with relatively high costs for any $r \in R_0$, $c(r) = C \gg SP^{\max}$ (where $SP^{\max}$ is the length of the greatest minimum shortest path between any two nodes), and demands for any $r \in R_0$, $d(r) = \delta > 0$, linking the dummy nodes to v_0. Finally, the vehicles capacities should be increased to $D + 2\delta$.

A new graph $G_1(V_1,E_1)$ is then formed, where $V_1 = V \cup V_0$ and $E_1 = E \cup R_0$. This transformation has complexity $O(M)$, and assuming $M < |R|$ (Property 1), the reduction at hand is linear with respect to the size of G.

Consider L_G^* and $P_{G_1}^*$ the induced graphs by CARP and OCARP optimal solutions in G and G_1, respectively. The relationship between them is described in the following:

1. $L_G^* = P_{G_1}^* \backslash V_0$, i.e., a CARP optimal solution is obtained removing the dummy nodes and edges from the corresponding OCARP optimal solution.
2. $c(L_G^*) = c(P_{G_1}^*) - 2MC$, i.e., the CARP optimal solution cost is equal to the corresponding OCARP optimal solution cost subtracting the costs of the dummy edges.

To prove Theorem 1 it will be demonstrated that in an OCARP optimal solution all tours traverse exactly two dummy edges by starting and ending at distinct dummy nodes (Lemma 5.1), and after extracting all dummy edges from this solution, a feasible CARP solution emerge (Corollary 5.1), i.e., a solution which traverses all required edges, attends the vehicles capacities (Lemma 5.2), and is formed by closed tours which visit the depot (Lemma 5.3). The proof is complete when, beyond feasibility, optimality is also verified (Lemma 5.4). For the following, consider L_G as the induced graph from an OCARP optimal solution after removing dummy nodes and edges, i.e., $L_G = P_{G_1}^* \backslash V_0$.

Lemma 5.1. *All tours from $P_{G_1}^*$ traverse exactly two distinct dummy edges.*

Proof. Since the transformed OCARP instance has M vehicles and $2M$ dummy edges, then two possibilities arise: (1) all tours traverse exactly two dummy edges or (2) at least one tour traverses more than two dummy edges. In the first case, two dummy edges can be traversed by a single tour without the requirement of revisiting any dummy edge. This can be accomplished if and only if the tour starts at a dummy node and ends at

another dummy node. In the second case, if a tour visits more than two dummy edges, at least one of them will necessarily be revisited, which would result in a needless cost increase. Therefore the second possibility could not occur in an optimal solution. □

Lemma 5.2. *All required edges from G are serviced in L_G without overloading any vehicle capacity.*

Proof. Since L_G is formed by extracting two dummy edges with demand δ from each $P^*_{G_1}$ OCARP tour, then naturally L_G traverses all required edges from G. It should be noticed that the vehicles remaining capacities are exactly the same for both solutions, since the original vehicles have 2δ less capacity than the upgraded vehicles. □

Lemma 5.3. *L_G is formed by a set of closed tours that visit the depot v_0.*

Proof. It was claimed by the proof of Lemma 5.1 that all $P^*_{G_1}$ OCARP tours have two distinct dummy edges as terminals. Since all dummy edges are linked to G by the depot v_0, removing them transforms the OCARP tours into cycles that visit v_0. □

Corollary 5.1. *L_G is feasible for the CARP.*

Note. As an immediate consequence of the previous lemmas, L_G does not overload any vehicle capacity, traverses all the required edges, and it is formed by closed tours that visit the depot v_0.

Lemma 5.4. *L_G represents an optimal CARP solution.*

Proof. The OCARP optimal solution cost for G_1 can be decomposed as $c(P^*_{G_1}) = c(P^*_{G_1} \setminus V_0) + 2MC$, where $2MC$ represents a lower bound. It follows that $c(P^*_{G_1} \setminus V_0)$ must be minimum, and so is $c(L_G)$, according with the hypothesis $L_G = P^*_{G_1} \setminus V_0$. □

Corollary 5.2. *OCARP is NP-hard.*

Note. CARP is an NP-hard problem [1] which is polynomially reducible to OCARP.

6. Solving strategy

This section reveals an inverse polynomial reduction of the one proposed in Section 5, i.e., OCARP $\leq_p$ CARP. This is relevant, once it admits solving OCARP through CARP algorithms.

Theorem 2. *The OCARP can be reduced polynomially into CARP.*

Consider an OCARP instance $G(V,E)$ with M vehicles. Add a dummy node v_0 and a set of dummy non-required edges (E_0), with relatively high costs for any $e \in E_0$, $c(e) = C \gg SP^{\max}$, and demands for any $e \in E_0$, $d(e) = 0$, linking v_0 to every node in G. A new graph $G_1(V_1, E_1)$ is thus formed, where $V_1 = V \cup \{v_0\}$ and $E_1 = E \cup E_0$. This reduction has linear complexity $O(|V|)$.

Consider P^*_G and $L^*_{G_1}$ the induced graphs by OCARP and CARP optimal solutions in G and G_1, respectively. The relationship between them is described in the following:

1. $P^*_G = L^*_{G_1} \setminus \{v_0\}$, i.e., an OCARP optimal solution is obtained by extracting the dummy edges from the corresponding CARP optimal solution.
2. $c(P^*_G) = c(L^*_{G_1}) - 2MC$, i.e., the OCARP optimal solution cost is equal to the corresponding CARP optimal solution cost subtracting twice the number of vehicles times the cost of a dummy edge.

Proof. Consider P_G as the induced graph from a CARP optimal solution after removing dummy node and edges, i.e., $P_G = L^*_{G_1} \setminus \{v_0\}$. Given the simplicity of this transformation, it will be demonstrated that P_G represents a feasible OCARP solution and conclude that it is also optimal.

An OCARP solution is considered feasible if it is formed by tours (open or not) that traverse all required edges without overloading the capacity of any vehicle. The induced graph P_G is formed by extracting only dummy edges from $L^*_{G_1}$, therefore P_G tours still traverse all the original required edges while respecting the capacity constraints. In addition, since the depot is linked to V only through dummy edges, then every $L^*_{G_1}$ tour contains exactly two dummy edges. If two adjacent edges are extracted from a cycle, as in every $L^*_{G_1}$ tour to generate P_G, the result will be an OCARP tour. This concludes P_G feasibility. Now about P_G being optimal, suppose there exists P^*_G, such that $c(P^*_G) < c(P_G)$. Then by simply adding two dummy edges, with cost C, at both beginning and end of each tour, it would be possible to transform P^*_G into an induced CARP solution graph, L_{G_1}, with cost $c(L_{G_1}) = c(P^*_G) + 2MC < c(P_G) + 2MC = c(L^*_{G_1})$, which would be a contradiction. □

From Property 2, Theorem 2 remains valid even if the set E_0 is the set of dummy edges linking v_0 and the terminal nodes from edges in R. This gives a simplification for the proposed reduction.

Corollary 6.1. *OCARP and CARP are complexity equivalent.*

Note. Since both polynomial reductions OCARP $\leq_p$ CARP and CARP $\leq_p$ OCARP exist, then these problems are equally hard to solve.

7. CARP path-scanning heuristics

7.1. General concepts

The path-scanning heuristics developed for CARP construct each solution by adding to a path starting at the depot, one required edge at a time. To determine the next edge to add, an *edge-selection rule* $\psi(e)$ is used (18), where $e = [v_i, v_j]$ is a candidate for the next required edge to be visited, v_l is the last node visited by the tour, and SP represents the shortest path distances between two nodes. Every unserviced required edge whose demand d_{ij} is less than the vehicle remaining capacity is a possible candidate, and the heuristic will choose the one which minimizes $\psi(e)$

$$\psi(e) = \min(SP(v_l, v_i), SP(v_l, v_j)). \tag{18}$$

There are cases where more than one candidate edge minimizes $\psi(e)$, more occasionally when they are incident to v_l. In these situations, a tie breaking rule is considered, and this rule represents the major difference between CARP path-scanning heuristics. Golden et al. [1] have used five criteria to break ties:

1. minimize c_{ij}/d_{ij}.
2. maximize c_{ij}/d_{ij}.
3. minimize the cost back to depot.
4. maximize the cost back to depot.
5. criterion 3 if the vehicle has used more than half of its capacity; criterion 4, otherwise.

A problem instance is solved five times, using a different criterion each time, and the best of the five solutions is taken. Pearn [33] modified this approach by selecting one of the five criteria at random, with equal probability, whenever a tie occurs. Belenguer et al. [20] simplified the tie breaking rule by randomly selecting one tied edge. This was copied by Santos et al. [3],

in their path-scanning heuristic with ellipse rule, explained in the following section.

7.2. Path-scanning heuristic with ellipse rule

Recently, Santos et al. [3] developed the so far best CARP path-scanning heuristic which makes use of an *ellipse rule*. When a vehicle is near its full capacity, this rule enforces the vehicle to service only edges near the shortest path between the last serviced edge and the depot, following the rationale that a heavily loaded vehicle should stay closer to the depot in order to reduce its returning cost. These authors define $ned = |R|$, td the total demand to be serviced, tc the total cost from edges with positive demand, v_0 the depot node, $[v_h,v_l]$ the last serviced edge on the tour, and β a real parameter. If the remaining vehicle capacity is less than or equal to $\beta(td/ned)$, then the next edge to be serviced $[v_i, v_j]$ must be the nearest edge to $[v_h, v_l]$ ($v_l = v_i$, if the edges are adjacent) satisfying the condition:

$$SP(v_l,v_i)+c_{ij}+SP(v_j,v_0) \leq \frac{tc}{ned}+SP(v_l,v_0). \quad (19)$$

If no candidate edge satisfies (19) then the vehicle returns to the depot. Through the ellipse rule, the authors obtained 44% reduction in overall average deviation from lower bounds with little or no increase in solution time, compared to previous path-scanning heuristics.

Solving OCARP using the path-scanning heuristic with ellipse rule was attempted. This, however, was not successful given that this heuristic was developed for the CARP version where the number of vehicles is a decision variable (Section 1), and in OCARP the number of vehicles must be a fixed parameter (Property 1). In fact, the computational tests (Section 8) reveal that this heuristic has failed to find even a single feasible solution for some instances. The main reason for this is that underneath the tour cost optimization problem relies a bin-packing subproblem of assigning the required edges among the vehicles, given their limited amount and capacity. Surely OCARP heuristics must take both these optimization problems into consideration, and thus a reactive path-scanning heuristic with ellipse rule is proposed, subject of the following.

7.3. Reactive path-scanning heuristic with ellipse rule

Two new features were introduced in this reactive version of the path-scanning heuristic with ellipse rule, and they will be described next:

7.3.1. Cost-demand edge-selection rule

One well-known bin-packing algorithm [34] is the *first fit decreasing* (FFD) that operates by sorting the elements in a decreasing order by demand, and then inserting each element into the first bin with sufficient remaining capacity. This concept of prioritizing the elements of higher demands inspired a new edge-selection rule $\tilde{\psi}(e)$ (20), redesigned in order to consider not only the shortest path cost, but also the demands to be collected from the candidate edges. These two objectives are weighted through parameter $\gamma \in [0,1]$, $e=[v_i,v_j]$ is a candidate edge, v_l is the last node visited by the tour, $SP^{\max}$ is the length of the greatest minimum shortest path and $d^{\max}$ is the maximum edge demand:

$$\tilde{\psi}(e) = \gamma \frac{\min(SP(v_l,v_i),SP(v_l,v_j))}{SP^{\max}} + (1-\gamma)\left(1-\frac{d_{ij}}{d^{\max}}\right). \quad (20)$$

It should be noticed that if $\gamma = 1$, then this edge-selection rule becomes equivalent to (18) (except by the normalization factor $SP^{\max}$). The parameter γ is self-tuned through a metaparameter $g \in [0.5,1)$, according to the feasibility of the previous solutions obtained by the heuristic. Let γ_k and g_k be the values of these parameters at iteration k. Table 1 gives the tuning mechanism for these parameters. The values of γ and g are set so that initially $\tilde{\psi}(e)$ works as in the original heuristic. When infeasible solutions are obtained in sequel, an exponentially fast adjustment of γ is performed by successive multiplications with g. During this adjustment, when a feasible solution appears, γ is readjusted in favor of solution cost and g is taken closer to 1 making future adjustments increasingly smoother.

Table 1
Tuning scheme for the parameters γ and g.

Initial values	Feasible solution at iteration k	Infeasible solution at iteration k
$\gamma_1 = 1$	$\gamma_{(k+1)} = \frac{2\gamma_k+1}{3}$	$\gamma_{(k+1)} = g_k\gamma_k$
$g_1 = 0.5$	$g_{(k+1)} = \frac{9g_k+1}{10}$	$g_{(k+1)} = g_k$

7.3.2. Reactive parameter β

The parameter β, as seen in Section 7.2, is responsible for controlling the ellipse rule shape, or in other words, how often will it be activated. Santos et al. [3] tried several values to find out that $\beta = 1.5$ was the best option for CARP. In this work, however, after trying several values for β, none of them were suitable for all instances. While lower values were more fitting to tighter instances, higher values seemed better for looser ones (details in Section 8). A reactive scheme, based on the work of Prais and Ribeiro [35], was implemented to select the value of β at each iteration of the heuristic from a discrete set of possible values. This selection is guided by the average quality of the solutions previously produced by each value. Let $B = \{\beta_0,\beta_1,\ldots,\beta_n\}$ be the set of possible values for β. The probabilities associated with the choice of each value are all initially made equal to $p_i = 1/n$, $(i = 1,\ldots,n)$. Furthermore, let c_{best} be the cost of the incumbent best solution and A_i the average cost of all solutions obtained by using $\beta = \beta_i$. The selection probabilities are periodically reevaluated through (21). In the reactive path-scanning algorithm, the possible β values were set in $B = \{0.0,0.5,1.0,1.5,2.0\}$

$$p_i = \frac{q_i}{\sum_{j=1}^{n} q_j}, \quad q_i = \frac{c_{best}}{A_i} \ (i = 1,\ldots,n). \quad (21)$$

Values of β_i producing good solutions on average will generate larger q_i, which in turn increases the probabilities p_i associated to them.

7.3.3. Heuristic pseudo-code

A pseudo-code to the reactive path-scanning heuristic with ellipse rule is given in Algorithm 3. This heuristic has a main loop, where a solution, not always feasible, is constructed and compared with the incumbent best in each iteration. Two procedures, `reactiveChoice()` (Algorithm 1) and `solConstruction()` (Algorithm 2), are used by the heuristic. The first procedure stochastically chooses a value for parameter β from set B, based on the average solution cost A_i obtained by each $\beta = \beta_i$. Until at least one solution ($N_i \geq 1$) has been built with $\beta = \beta_i$, the corresponding A_i is not used to compute q_i (21) (refer to previous section for a thorough description). The second procedure, which represents the core of the heuristic, is responsible for building a solution using the edge-selection (20) and the ellipse (19) rules. It is worth mentioning that this procedure, while inspired in CARP path-scanning heuristics, was adapted to OCARP by not starting (and ending) the tours at the depot. In Algorithm 3, set A is initialized with zeros (line 3) for computational purpose, and it is further updated at line 27.

The computational complexity of the reactive path-scanning heuristic with ellipse rule is mainly governed by procedure

`solConstruction()`, which evaluates all unserviced required edges to decide the next candidate edge to become part of the solution, this being repeated $|R|$ times. The complexity of `solConstruction()` is thus bounded by $O(|R|^2)$, and since it is contained in the main loop, which executes $10^5\lceil\sqrt{|R|}\rceil$ iterations, the complexity of Algorithm 3 is bounded by $O(|R|^{5/2})$.

Algorithm 1. reactiveChoice(A,N,c_{best})

Input: A—average solution costs for each β, N—number of solutions obtained for each β, c_{best}—cost of the incumbent best solution
Output: i—index of B containing the chosen value for β
1: $q_{sum} \leftarrow 0$
2: **for** ($i=1$ **to** $|B|$) **do**
3: **if** $N_i \geq 1$ **then**
4: $q_i \leftarrow \frac{c_{best}}{A_i}$
5: **else**
6: $q_i \leftarrow 1.0$ // biased in favor of choosing $\beta = B_i$
7: **end if**
8: $q_{sum} \leftarrow q_{sum} + q_i$
9: **end for**
10: $n_{rand} \leftarrow$ **randomNumber**(0,1) // real random number between [0,1]
11: $p_{sum} \leftarrow 0$
12: **for** ($i=1$ **to** $|B|$)**do**
13: $p_{sum} \leftarrow p_{sum} + \frac{q_i}{q_{sum}}$
14: **if** $n_{rand} \leq p_{sum}$ **then**
15: **break for**
16: **end if**
17: **end for**
18: **return** i

Algorithm 2. solConstruction (G,M,D,β,γ)

Input: G—instance graph, M—number of vehicles available, D—vehicle capacity, β—ellipse rule parameter, γ—cost-demand edge-selection rule parameter
Output: S—set of OCARP tours
1: $td = \sum_{e \in R} d(e)$, $ned = |R|$
2: $S \leftarrow \varnothing$
3: $t \leftarrow 1$ // tour index
4: $tour_1 \leftarrow \varnothing$ // ordered set of edges representing a tour
5: $rvc \leftarrow D$ // remaining vehicle capacity
6: **for** ($i=1$ **to** ned) **do**
7: **if** $t > M$ **then**
8: // number of tours exceeds M
9: $S \leftarrow \varnothing$
10: **break for**
11: **end if**
12: $F \leftarrow \text{argmin}_{e \in R \setminus S} \tilde{\psi}(e)$ // set of candidates edges minimizing $\tilde{\psi}(e)$
13: **if** $rvc \leq \beta \frac{td}{ned}$ **then**
14: $F \leftarrow F \setminus$ {edges violating (19)}
15: **end if**
16: **if** $F = \varnothing$ **then**
17: // starting new tour
18: $S \leftarrow S \cup \{tour_t\}$
19: $t \leftarrow t+1$
20: $tour_t \leftarrow \varnothing$
21: $rvc \leftarrow D$
22: **else**
23: $e \leftarrow$ **randomEdge**(F) // randomly selects an edge from set of candidates
24: $tour_t \leftarrow tour_t \cup \{e\}$
25: $rvc \leftarrow rvc - d(e)$
26: **end if**
27: **end for**
28: **return** S

Algorithm 3. Reactive Path-Scanning Heuristic with Ellipse Rule

Input: $G(V,E)$—instance graph, M—number of vehicles available, D—vehicle capacity
Output: S_{best} is the feasible solution with the lowest cost obtained
1: $S_{best} \leftarrow \varnothing$
2: $c_{best} \leftarrow 2\sum_{e \in E} c(e)$, $c_{worst} \leftarrow \sum_{e \in R} c(e)$ // trivial upper and lower bounds
3: $A \leftarrow \{0.0,0.0,0.0,0.0,0.0\}$ // average solution cost for each β
4: $B \leftarrow \{0.0,0.5,1.0,1.5,2.0\}$ // possible values for β
5: $N \leftarrow \{0,0,0,0,0\}$ // number of solutions obtained for each β
6: $\gamma \leftarrow 1.0$, $g \leftarrow 0.5$, $maxIter \leftarrow 10^5\lceil\sqrt{|R|}\rceil$, $k = 1$
7: **while** ($k \leq maxIter$) **do**
8: $i \leftarrow$ **reactiveChoice**(A,N,c_{best}) // see Algorithm 1
9: $\beta \leftarrow B_i$
10: $S \leftarrow$ **solConstruction**(G,M,D,β,γ) // see Algorithm 2
11: **if** ($S \neq \varnothing$) **then**
12: // S is a feasible OCARP solution
13: $c_{sol} \leftarrow$ **cost**(S) // get S cost
14: **if** ($c_{sol} < c_{best}$) **then**
15: $S_{best} \leftarrow S$
16: $c_{best} \leftarrow c_{sol}$
17: **else if** ($c_{sol} > c_{worst}$) **then**
18: $c_{worst} \leftarrow c_{sol}$
19: **end if**
20: $\gamma \leftarrow \frac{2\gamma+1}{3}$
21: $g \leftarrow \frac{9g+1}{10}$
22: **else**
23: // S is OCARP infeasible
24: $\gamma \leftarrow g\gamma$
25: $c_{sol} \leftarrow c_{worst}$ // penalizing cost of infeasible solutions
26: **end if**
27: $N_i \leftarrow N_i + 1$
28: $A_i \leftarrow A_i + \frac{c_{sol} - A_i}{N_i}$ // updating the average solution cost
29: $k \leftarrow k+1$
30: **end while**
31: **return** S_{best}

8. Computational experiments

Having in mind that OCARP is a new NP-hard combinatorial optimization problem, the objective of these computational experiments consisted in forming a new set of instances and conferring the first lower and upper bounds to the optimal costs.

To form the set of OCARP instances the standard set of CARP instances[1] was referred to, which includes 23 *gdb* (7–27 nodes, 11–55 edges) [2], 34 *val* (24–50 nodes, 34–97 edges) [17], 24 *egl* (77–140 nodes, 98–190 edges) [36], 32 *A*, and 24 *B* instances (10–40 nodes, 15–69 edges) [25], totaling 137 instances. The depot was considered a common node while the rest of the data left intact, leading to five groups of instances referred as *ogdb*, *oval*, *oegl*, *oA*, and *oB*.

Since OCARP is only meaningful with a fixed M (Property 1), the computational tests have considered three classes of

[1] http://www.uv.es/~belengue/carp.html; http://www.hha.dk/~sanw

parameterization: $M = M^*$, $M = M^*+1$ and $M = M^*+2$, where M^* represents the minimum number of vehicles necessary for a feasible solution. Consequently, from each of the 137 CARP instances, three different numbers of vehicles are considered, thus deriving 411 OCARP instances.

All tests were executed in a Intel Core 2 Quad 3.0 GHz with 4 Gb of RAM.

8.1. Lower bounds

A trivial lower bound for an optimal OCARP solution cost may be computed by summing all required edges costs $LB_0 = \sum_{e \in R} c_e$. This, however, can be very loose for those instances which require deadheading. Welz [27] has observed that the linear relaxation of the CARP model, excluding the subtour elimination constraints (6), always provides a lower bound equal to LB_0. This is extendable to OCARP, i.e., the linear relaxation of the OCARP model, excluding the subtour elimination constraints (16), generates a lower bound equal to LB_0 (the proof is equivalent to the one given for CARP [27]).

Lower bounding schemes for CARP, such as *Capacity constraints*, *Odd Edge Cutset constraints*, *Disjoint Paths* [18], and *Multiple Cuts Node Duplication Lower Bound* [19] affirm that, for a subset $S \subseteq V \backslash \{v_0\}$, some edges must be deadheaded depending on the total demand to be serviced in S, the number of required edges in the cutset $(S, S\backslash V)$, and possibly some additional demand on the path from v_0 to S. However, these constraints are valid based on the fact that every CARP vehicle which enters subset S must exit from it to return to the depot. Since OCARP tours may start and end at any node, these inequalities are not valid for OCARP.

Two attempts were made to improve LB_0 using the optimization software CPLEX 12. The first one was by solving OCARP linear relaxation model including only part ($S \subseteq V, |S| \leq 3$) of the subtour elimination constraints (16). This strategy, however, did not improve LB_0. The second approach consisted in solving the integer reduced OCARP model, neglecting all subtour elimination constraints. The stopping criteria were defined by execution time (1 h), memory usage (2.5 Gb) or optimality. When one of these criteria was reached, the best lower bound obtained in the process was stored. These lowerbounds (*LB*) are available in Table 3.

8.2. Upper bounds

The reactive path-scanning heuristic with ellipse rule (RPS_ER) was compared with the path-scanning heuristics from Belenguer et al. [20] (PS) and Santos et al. [3] (PS_ER), the last one considering three configurations for the ellipse rule parameter β. These heuristics were implemented in C programming language, and executed for $10^5 \lceil \sqrt{|R|} \rceil$ iterations over the testbench of 411 instances, divided into five groups, *ogdb*, *oval*, *oegl*, *oA*, *oB*; and three classes, M^*, M^*+1, M^*+2 (Table 2).

When considering each group–class of instances (Table 2), the RPS_ER heuristic achieved the smallest $\Delta LB = (UB-LB)/LB$ for all except one group–class, namely *ogdb*–(M^*+2). In addition, RPS_ER has also achieved the lowest $\Delta LB^{\max}$ for six group–classes, which is at least two times better than the other heuristics. In the overall comparison, RPS_ER outperformed its siblings for all three classes regarding both ΔLB and $\Delta LB^{\max}$.

As mentioned in Section 7.2, heuristic PS_ER was unable to find feasible solutions for some instances of several group–classes (represented by a dash in Table 2). This effect is amplified for higher values of β, since it shortens the tours more prematurely. Nevertheless, the looser instances for which PS_ER did find feasible solutions had their ΔLB decreased as β increased, meaning that the ellipse rule behaves more effectively for these group–classes. A glimpse on the hardness to attain a feasible solution for each instance group–class is given by *Feas*, which represents the percentage of feasible solutions obtained compared to the total number of iterations.

Table 3 shows the individual results for all instances after running $10^5 \lceil \sqrt{|R|} \rceil$ iterations of RPS_ER. From the set of 411 instances, 133 solutions (32.36%) were proven optimal ($LB = UB$). The reduced integer model lower bound (*LB*) improved the lower bounds of 59 instances, which corresponds to 20.27% of the 291

Table 2
Comparison results between path-scanning heuristics.

Group	Class	PS			PS_ER									RPS_ER		
					$\beta = 0.5$			$\beta = 1.0$			$\beta = 1.5$					
		ΔLB	$\Delta LB^{\max}$	Feas	ΔLB	$\Delta LB^{\max}$	Feas	ΔLB	$\Delta LB^{\max}$	Feas	ΔLB	$\Delta LB^{\max}$	Feas	ΔLB	$\Delta LB^{\max}$	Feas
ogdb	M^*	0.50	5.48	94.96	0.38	3.33	90.34	**0.29**	2.74	77.82	1,90	17.29	62.01	**0.29**	**2.38**	77.43
	M^*+1	0.48	5.02	100.00	0.36	3.81	100.00	0.22	**1.90**	100.00	**0.20**	**1.90**	99.00	**0.20**	2.25	98,44
	M^*+2	0.48	5.02	100.00	0.36	3.81	100.00	0.22	1.90	100.00	**0.13**	**1.69**	100.00	0.20	2.25	99.95
oval	M^*	5.12	**13.70**	94,36	4.91	29.45	91.42	–	–	84.48	–	–	77.40	**4.22**	18.49	83.65
	M^*+1	5.93	9.71	100.00	5.32	9.71	100.00	4.50	9.35	99.91	4.43	9.35	97.39	**4.21**	**8.27**	97.96
	M^*+2	6.46	10.07	100.00	5.85	10.07	100.00	5.01	9.35	100.00	4.91	9.35	100.00	**4.67**	**9.23**	99.70
oegl	M^*	53.24	132.85	82.87	–	–	69.58	–	–	34.99	–	–	9.66	**42.76**	**115.77**	49.07
	M^*+1	48.18	71.95	98.69	34.77	58.32	95.05	30.99	83.80	81.88	–	–	61.80	**28.41**	**40.82**	80.65
	M^*+2	47.08	71.95	100.00	34.33	58.32	99.88	27.73	48.64	95.32	25.38	**36.48**	83.45	**25.21**	39.60	90.73
oA	M^*	8.12	**65.05**	80.02	15.71	165.65	72.64	–	–	61.06	–	–	53.79	**7.43**	**65.05**	65.08
	M^*+1	7.40	44.98	94.93	5.52	**40.43**	93.13	–	–	85.54	–	–	74.13	**5.20**	47.72	84.44
	M^*+2	7.50	29.18	96.85	5.62	**17.33**	96.41	5.41	41.64	93.65	–	–	85.69	**4.73**	24.62	90.52
oB	M^*	8.89	**29.20**	74.44	8.23	45.26	70.32	–	–	49.28	–	–	39.34	**8.02**	34.46	55.54
	M^*+1	6.67	22.10	93.51	4.54	**14.98**	90.15	4.10	21.72	83.21	–	–	70.78	**3.50**	16.10	79.63
	M^*+2	6.60	19.10	99.18	4.25	13.48	97.30	3.14	**9.36**	88.86	2.92	13.50	83.43	**2.75**	10.11	88.22
overall	M^*	14.13	132.85	85.61	–	–	79.33	–	–	63.05	–	–	50.77	**11.73**	**115.77**	67.28
	M^*+1	12.89	71.95	97.45	9.56	58.32	95.80	–	–	90.48	–	–	81.33	**7.88**	**47.72**	88.64
	M^*+2	12.84	71.95	99.12	9.58	58.32	98.67	7.95	48.64	95.75	–	–	90.85	**7.20**	**39.60**	94.01

PS, path-scanning heuristic with random selection of tied edges [20]; PS_ER, path-scanning heuristic with ellipse rule [3].
RPS_ER, reactive path-scanning heuristic with ellipse rule; β, ellipse rule parameter; M, maximum number of vehicles.
ΔLB, average deviation from lower bound (%); $\Delta LB^{\max}$, maximum average deviation from lower bound (%); *Feas*, average number of feasible solutions (%).

Table 3
Results for the reactive path-scanning heuristic with ellipse rule.

	M^*	LB_0	M^*					M^*+1					M^*+2				
			LB	UB	ΔLB	CPU	Feas	LB	UB	ΔLB	CPU	Feas	LB	UB	ΔLB	CPU	Feas
ogdb1	5	252	252	**252**	0.00	0.00	97.80	252	**252**	0.00	0.00	100.00	252	**252**	0.00	0.00	100.00
ogdb2	6	291	291	**291**	0.00	0.00	100.00	291	**291**	0.00	0.00	100.00	291	**291**	0.00	0.00	100.00
ogdb3	5	233	233	**233**	0.00	0.00	96.77	233	**233**	0.00	0.00	100.00	233	**233**	0.00	0.00	100.00
ogdb4	4	238	238	**238**	0.00	0.00	78.57	238	**238**	0.00	0.00	100.00	238	**238**	0.00	0.00	100.00
ogdb5	6	316	316	**316**	0.00	0.00	98.04	316	**316**	0.00	0.00	100.00	316	**316**	0.00	0.00	100.00
ogdb6	5	260	260	**260**	0.00	0.00	97.56	260	**260**	0.00	0.00	100.00	260	**260**	0.00	0.00	100.00
ogdb7	5	262	262	**262**	0.00	0.00	97.06	262	**262**	0.00	0.00	100.00	262	**262**	0.00	0.00	100.00
ogdb8	10	210	210	215	2.38	1.02	59.63	210	214	1.90	1.18	87.34	210	213	1.43	1.22	99.60
ogdb9	10	219	219	224	2.28	1.31	48.94	219	220	0.46	1.56	84.06	219	221	0.91	1.63	99.36
ogdb10	4	252	252	**252**	0.00	0.01	90.82	252	**252**	0.00	0.00	100.00	252	**252**	0.00	0.00	100.00
ogdb11	5	356	356	363	1.97	0.90	98.71	356	364	2.25	0.90	100.00	356	364	2.25	0.91	100.00
ogdb12	7	336	336	**336**	0.00	0.00	58.62	336	**336**	0.00	0.00	95.04	336	**336**	0.00	0.00	100.00
ogdb13	6	509	509	**509**	0.00	0.15	1.78	509	**509**	0.00	0.00	98.04	509	**509**	0.00	0.00	100.00
ogdb14	5	96	96	**96**	0.00	0.00	100.00	96	**96**	0.00	0.00	100.00	96	**96**	0.00	0.00	100.00
ogdb15	4	56	56	**56**	0.00	0.00	100.00	56	**56**	0.00	0.00	100.00	56	**56**	0.00	0.00	100.00
ogdb16	5	119	119	**119**	0.00	0.00	62.07	119	**119**	0.00	0.00	100.00	119	**119**	0.00	0.00	100.00
ogdb17	5	84	84	**84**	0.00	0.00	100.00	84	**84**	0.00	0.00	100.00	84	**84**	0.00	0.00	100.00
ogdb18	5	158	158	**158**	0.00	0.00	100.00	158	**158**	0.00	0.00	100.00	158	**158**	0.00	0.00	100.00
ogdb19	3	45	45	**45**	0.00	0.00	91.48	45	**45**	0.00	0.00	100.00	45	**45**	0.00	0.00	100.00
ogdb20	4	105	105	**105**	0.00	0.02	24.84	105	**105**	0.00	0.00	100.00	105	**105**	0.00	0.00	100.00
ogdb21	6	149	149	**149**	0.00	0.01	61.49	149	**149**	0.00	0.00	100.00	149	**149**	0.00	0.00	100.00
ogdb22	8	191	191	**191**	0.00	0.09	79.90	191	**191**	0.00	0.09	99.98	191	**191**	0.00	0.09	100.00
ogdb23	10	223	223	**223**	0.00	0.40	36.73	223	**223**	0.00	0.02	99.74	223	**223**	0.00	0.02	100.00
oval1A	2	146	154	**154**	0.00	0.63	100.00	150	154	2.67	0.63	100.00	146	154	5.48	0.63	100.00
oval1B	3	146	148	154	4.05	0.51	30.63	146	149	2.05	0.70	100.00	146	149	2.05	0.70	100.00
oval1C	8	146	146	173	18.49	0.89	0.72	146	149	2.05	0.89	75.02	146	**146**	0.00	0.09	94.60
oval2A	2	185	195	**195**	0.00	0.46	100.00	191	195	2.09	0.47	100.00	185	195	5.41	0.47	100.00
oval2B	3	185	191	192	0.52	0.52	100.00	185	192	3.78	0.50	100.00	185	192	3.78	0.50	100.00
oval2C	8	185	185	197	6.49	0.68	14.11	185	186	0.54	0.66	80.37	185	**185**	0.00	0.11	96.47
oval3A	2	65	71	**71**	0.00	0.48	100.00	68	71	4.41	0.47	100.00	65	71	9.23	0.49	100.00
oval3B	3	65	69	**69**	0.00	0.51	96.32	65	68	4.62	0.52	100.00	65	68	4.62	0.52	100.00
oval3C	7	65	65	68	4.62	0.45	36.78	65	**65**	0.00	0.49	84.40	65	**65**	0.00	0.41	98.65
oval4A	3	343	353	361	2.27	1.69	99.94	343	365	6.41	1.64	100.00	344	365	6.10	1.67	100.00
oval4B	4	343	343	363	5.83	1.74	99.58	343	363	5.83	1.70	100.00	343	363	5.83	1.78	100.00
oval4C	5	343	343	364	6.12	1.74	73.23	343	355	3.50	1.86	100.00	343	355	3.50	1.89	100.00
oval4D	9	343	343	359	4.66	2.18	79.00	343	354	3.21	2.24	99.90	343	359	4.66	2.25	100.00
oval5A	3	367	375	387	3.20	1.55	99.97	367	383	4.36	1.47	100.00	367	383	4.36	1.52	100.00
oval5B	4	367	368	385	4.62	1.56	99.38	367	386	5.18	1.64	100.00	367	386	5.18	1.61	100.00
oval5C	5	367	368	381	3.53	1.67	90.98	367	378	3.00	1.76	100.00	367	378	3.00	1.70	100.00
oval5D	9	367	367	377	2.72	2.04	91.10	367	378	3.00	2.03	100.00	367	378	3.00	2.04	100.00
oval6A	3	190	194	196	1.03	1.01	100.00	190	196	3.16	0.99	100.00	190	196	3.16	1.00	100.00
oval6B	4	190	190	197	3.68	1.03	87.45	190	194	2.11	1.08	100.00	190	194	2.11	1.07	100.00
oval6C	10	190	190	195	2.63	1.32	72.36	190	192	1.05	1.44	94.63	190	192	1.05	1.44	99.94
oval7A	3	249	256	267	4.30	1.54	99.94	249	267	7.23	1.48	100.00	249	267	7.23	1.54	100.00
oval7B	4	249	249	259	4.02	1.61	98.31	249	262	5.22	1.63	100.00	249	262	5.22	1.62	100.00
oval7C	9	249	249	262	5.22	1.69	54.60	249	261	4.82	2.11	97.75	249	256	2.81	2.07	100.00
oval8A	3	347	364	366	0.55	1.29	99.36	348	366	5.17	1.29	100.00	347	366	5.48	1.29	100.00
oval8B	4	347	347	364	4.90	1.37	96.76	347	360	3.75	1.34	100.00	347	360	3.75	1.38	100.00
oval8C	9	347	347	361	4.03	1.37	48.67	347	353	1.73	1.77	98.53	347	355	2.31	1.73	100.00
oval9A	3	278	292	303	3.77	2.48	100.00	290	303	4.48	2.48	100.00	278	303	8.99	2.57	100.00
oval9B	4	278	291	304	4.47	2.61	99.98	278	301	8.27	2.65	100.00	278	301	8.27	2.75	100.00
oval9C	5	278	287	304	5.92	2.72	99.78	278	297	6.83	2.81	100.00	278	297	6.83	2.79	100.00
oval9D	10	278	278	300	7.91	3.36	88.35	278	294	5.76	3.48	100.00	278	294	5.76	3.37	100.00
oval10A	3	376	394	409	3.81	2.54	100.00	385	409	6.23	2.55	100.00	376	409	8.78	2.41	100.00
oval10B	4	376	382	406	6.28	2.68	100.00	376	406	7.98	2.71	100.00	376	406	7.98	2.26	100.00
oval10C	5	376	376	406	7.98	2.85	99.80	376	404	7.45	2.81	100.00	376	404	7.45	2.34	100.00

Table 3 (*continued*)

	M^*	LB_0	M^*					M^*+1					M^*+2				
			LB	UB	ΔLB	CPU	Feas	LB	UB	ΔLB	CPU	Feas	LB	UB	ΔLB	CPU	Feas
oval10D	10	376	376	398	5.85	3.43	86.95	376	396	5.32	3.49	100.00	376	396	5.32	3.51	100.00
oegl-e1-A	5	1468	1595	1959	22.82	1.31	66.46	1538	1813	17.88	1.55	100.00	1513	1813	19.83	1.55	100.00
oegl-e1-B	7	1468	1492	1961	31.43	1.38	59.04	1478	1809	22.40	1.71	97.61	1468	1818	23.84	1.70	100.00
oegl-e1-C	10	1468	1468	1833	24.86	1.62	66.66	1468	1709	16.42	1.84	86.94	1468	1750	19.21	1.89	98.91
oegl-e2-A	7	1879	1955	2450	25.32	2.09	63.33	1895	2352	24.12	2.49	100.00	1879	2352	25.17	2.46	100.00
oegl-e2-B	10	1879	1879	2423	28.95	2.36	63.87	1879	2283	21.50	2.76	94.83	1879	2275	21.08	2.80	100.00
oegl-e2-C	14	1879	1879	2700	43.69	2.37	39.46	1879	2344	24.75	2.85	69.80	1879	2238	19.11	3.13	85.41
oegl-e3-A	8	2188	2198	3042	38.40	2.53	52.29	2188	2720	24.31	3.35	99.87	2188	2720	24.31	3.38	100.00
oegl-e3-B	12	2188	2188	2861	30.76	3.11	58.82	2188	2856	30.53	3.81	86.94	2188	2739	25.18	3.94	99.79
oegl-e3-C	17	2188	2188	3045	39.17	3.38	47.32	2188	2673	22.17	4.08	69.03	2188	2720	24.31	4.47	84.48
oegl-e4-A	9	2453	2453	3270	33.31	2.99	51.77	2453	3118	27.11	3.95	99.86	2453	2988	21.81	3.89	100.00
oegl-e4-B	14	2453	2453	3293	34.24	3.31	46.61	2453	3119	27.15	4.48	77.08	2453	3015	22.91	4.79	95.39
oegl-e4-C	19	2453	2453	4243	72.97	14.94	0.24	2453	3183	29.76	4.21	45.73	2453	3003	22.42	4.90	68.19
oegl-s1-A	7	1394	1517	2075	36.78	3.10	77.46	1394	1946	39.60	3.46	100.00	1394	1946	39.60	3.50	100.00
oegl-s1-B	10	1394	1394	1963	40.82	3.49	80.86	1394	1878	34.72	3.78	99.40	1394	1828	31.13	3.84	100.00
oegl-s1-C	14	1394	1394	2252	61.55	2.88	43.02	1394	1963	40.82	3.83	75.62	1394	1849	32.64	4.17	89.10
oegl-s2-A	14	3174	3174	4383	38.09	8.71	68.22	3174	4249	33.87	10.29	97.88	3174	4159	31.03	10.27	100.00
oegl-s2-B	20	3174	3174	5442	71.46	12.41	3.74	3174	4243	33.68	10.18	62.40	3174	4128	30.06	12.05	87.83
oegl-s2-C	27	3174	3174	5179	63.17	20.53	7.80	3174	4310	35.79	10.99	52.54	3174	4033	27.06	12.56	67.66
oegl-s3-A	15	3379	3379	4428	31.04	10.70	82.35	3379	4305	27.40	11.13	99.98	3379	4356	28.91	11.36	100.00
oegl-s3-B	22	3379	3379	4672	38.27	10.68	54.99	3379	4354	28.85	11.32	78.46	3379	4179	23.68	13.48	93.15
oegl-s3-C	29	3379	3379	4828	42.88	20.11	22.45	3379	4410	30.51	12.38	50.32	3379	4265	26.22	14.19	70.29
oegl-s4-A	19	4186	4186	5253	25.49	13.86	70.05	4186	5168	23.46	15.80	94.99	4186	5086	21.50	15.78	100.00
oegl-s4-B	27	4186	4186	5653	35.05	13.64	50.72	4186	5239	25.16	17.00	67.65	4186	4981	18.99	18.32	84.43
oegl-s4-C	35	4186	4186	9032	115.77	247.30	0.06	4186	5852	39.80	20.16	28.58	4186	5237	25.11	17.66	52.84
oAi10A	4	43	48	53	10.42	0.11	14.34	43	**43**	0.00	0.01	68.72	43	**43**	0.00	0.00	85.71
oAi10B	3	43	43	**43**	0.00	0.00	100.00	43	**43**	0.00	0.00	100.00	43	**43**	0.00	0.00	100.00
oAi10C	2	43	43	**43**	0.00	0.00	100.00	43	**43**	0.00	0.00	100.00	43	**43**	0.00	0.00	100.00
oAi10D	1	43	45	**45**	0.00	0.10	100.00	43	45	4.65	0.10	100.00	43	45	4.65	0.10	100.00
oAi13A	8	85	85	87	2.35	0.34	13.47	85	**85**	0.00	0.01	58.88	85	**85**	0.00	0.01	86.49
oAi13B	4	85	85	88	3.53	0.22	32.10	85	**85**	0.00	0.01	99.25	85	**85**	0.00	0.00	100.00
oAi13C	2	85	91	**91**	0.00	0.22	46.43	88	**88**	0.00	0.27	100.00	85	88	3.53	0.27	100.00
oAi13D	2	85	91	**91**	0.00	0.25	100.00	88	91	3.41	0.25	100.00	85	91	7.06	0.25	100.00
oAi15A	8	92	92	**92**	0.00	0.01	38.32	92	**92**	0.00	0.01	72.35	92	**92**	0.00	0.00	90.13
oAi15B	5	92	92	**92**	0.00	0.00	92.21	92	**92**	0.00	0.00	100.00	92	**92**	0.00	0.00	100.00
oAi15C	3	92	92	**92**	0.00	0.00	100.00	92	**92**	0.00	0.00	100.00	92	**92**	0.00	0.00	100.00
oAi15D	2	92	94	**94**	0.00	0.25	100.00	92	94	2.17	0.25	100.00	92	94	2.17	0.26	100.00
oAi20A	11	113	113	115	1.77	0.62	19.91	113	**113**	0.00	0.01	57.60	113	**113**	0.00	0.03	68.79
oAi20B	7	113	113	**113**	0.00	0.26	73.43	113	**113**	0.00	0.00	97.57	113	**113**	0.00	0.01	100.00
oAi20C	4	113	113	**113**	0.00	0.00	100.00	113	**113**	0.00	0.01	100.00	113	**113**	0.00	0.00	100.00
oAi20D	3	113	116	**116**	0.00	0.42	100.00	113	116	2.65	0.42	100.00	113	116	2.65	0.43	100.00
oAi24A	12	139	139	161	15.83	0.77	7.39	139	142	2.16	0.89	50.42	139	142	2.16	1.03	69.52
oAi24B	7	139	139	147	5.76	0.71	59.85	139	147	5.76	0.85	95.09	139	145	4.32	0.86	100.00
oAi24C	4	139	144	151	4.86	0.71	100.00	139	151	8.63	0.69	100.00	139	151	8.63	0.72	100.00
oAi24D	3	139	154	**154**	0.00	0.62	100.00	151	154	1.99	0.65	100.00	139	154	10.79	0.65	100.00
oAi27A	10	188	188	228	21.28	0.49	2.69	188	199	5.85	0.72	54.86	188	191	1.60	0.84	76.76
oAi27B	6	188	188	205	9.04	0.62	67.03	188	200	6.38	0.69	99.28	188	198	5.32	0.70	100.00
oAi27C	3	188	208	209	0.48	0.53	87.58	195	206	5.64	0.57	100.00	188	206	9.57	0.56	100.00
oAi27D	2	188	214	216	0.93	0.51	96.78	208	211	1.44	0.52	100.00	192	211	9.90	0.52	100.00
oAi31A	19	271	271	342	26.20	4.95	0.22	271	304	12.18	2.03	15.19	271	285	5.17	1.61	43.54
oAi31B	11	271	271	294	8.49	1.14	44.42	271	286	5.54	1.53	74.28	271	277	2.21	1.60	91.88
oAi31C	6	271	271	297	9.59	1.24	97.66	271	297	9.59	1.24	100.00	271	297	9.59	1.26	100.00
oAi31D	4	271	280	299	6.79	1.11	99.98	271	300	10.70	1.14	100.00	271	300	10.70	1.12	100.00
oAi40A	25	329	329	543	65.05	12.99	0.12	329	486	47.72	13.93	0.15	329	410	24.62	2.92	1.75
oAi40B	13	329	329	408	24.01	1.42	2.89	329	360	9.42	1.96	58.46	329	350	6.38	2.25	82.00
oAi40C	7	329	329	362	10.03	1.63	85.81	329	359	9.12	1.72	100.00	329	359	9.12	1.72	100.00
oAi40D	5	329	329	366	11.25	1.47	100.00	329	366	11.25	1.48	100.00	329	366	11.25	1.51	100.00

oBi10B	5	45	48	49	2.08	0.11	9.09	45	**45**	0.00	0.00	61.29	45	**45**	0.00	0.00	87.50
oBi10C	3	45	45	**45**	0.00	0.00	100.00	45	**45**	0.00	0.00	100.00	45	**45**	0.00	0.00	100.00
oBi10D	2	45	45	**45**	0.00	0.00	100.00	45	**45**	0.00	0.00	100.00	45	**45**	0.00	0.00	100.00
oBi13B	7	60	63	64	1.59	0.18	30.30	60	**60**	0.00	0.01	67.61	60	**60**	0.00	0.01	79.89
oBi13C	4	60	60	**60**	0.00	0.00	94.41	60	**60**	0.00	0.00	100.00	60	**60**	0.00	0.00	100.00
oBi13D	3	60	60	**60**	0.00	0.00	100.00	60	**60**	0.00	0.00	100.00	60	**60**	0.00	0.00	100.00
oBi15B	7	74	74	77	4.05	0.25	41.09	74	**74**	0.00	0.00	68.32	74	**74**	0.00	0.00	81.20
oBi15C	4	74	74	**74**	0.00	0.01	96.32	74	**74**	0.00	0.01	100.00	74	**74**	0.00	0.01	100.00
oBi15D	3	74	77	**77**	0.00	0.22	100.00	74	77	4.05	0.23	100.00	74	77	4.05	0.24	100.00
oBi20B	9	99	101	126	24.75	0.60	0.61	99	**99**	0.00	0.12	56.60	99	**99**	0.00	0.05	74.26
oBi20C	5	99	99	102	3.03	0.46	81.22	99	**99**	0.00	0.04	100.00	99	**99**	0.00	0.03	100.00
oBi20D	3	99	105	**105**	0.00	0.34	56.60	101	102	0.99	0.43	100.00	99	102	3.03	0.42	100.00
oBi24B	9	107	107	136	27.10	0.48	2.13	107	116	8.41	0.59	55.47	107	109	1.87	0.67	75.22
oBi24C	5	107	107	113	5.61	0.52	78.88	107	113	5.61	0.57	99.97	107	113	5.61	0.54	100.00
oBi24D	3	107	119	123	3.36	0.35	50.00	110	117	6.36	0.51	100.00	107	117	9.35	0.50	100.00
oBi27B	16	185	185	208	12.43	0.92	0.98	185	198	7.03	0.92	21.90	185	190	2.70	0.85	50.29
oBi27C	8	185	185	188	1.62	0.57	41.52	185	190	2.70	0.73	80.65	185	**185**	0.00	0.01	96.81
oBi27D	6	185	185	190	2.70	0.65	94.42	185	188	1.62	0.63	100.00	185	188	1.62	0.68	100.00
oBi31B	19	274	274	361	31.75	5.90	0.15	274	310	13.14	1.47	5.47	274	295	7.66	1.46	28.97
oBi31C	10	274	274	296	8.03	1.04	49.50	274	**274**	0.00	0.21	84.79	274	277	1.09	1.28	98.81
oBi31D	7	274	274	288	5.11	1.02	88.29	274	283	3.28	1.09	100.00	274	283	3.28	1.07	100.00
oBi40B	21	267	267	359	34.46	1.92	1.63	267	310	16.10	2.49	20.28	267	293	9.74	2.19	44.93
oBi40C	11	267	267	293	9.74	1.63	61.31	267	281	5.24	1.88	89.01	267	283	5.99	1.90	99.35
oBi40D	7	267	267	307	14.98	1.26	54.48	267	292	9.36	1.58	99.66	267	294	10.11	1.59	100.00

M^*, minimum number of vehicles to attain a feasible solution; LB_0, trivial lower bound; LB, reduced integer model lower bound; UB, reactive path-scanning heuristic with ellipse rule upper bound; ΔLB, average deviation from lower bound (%); CPU, running time (s) to attain UB; $Feas$, number of feasible solutions (%).

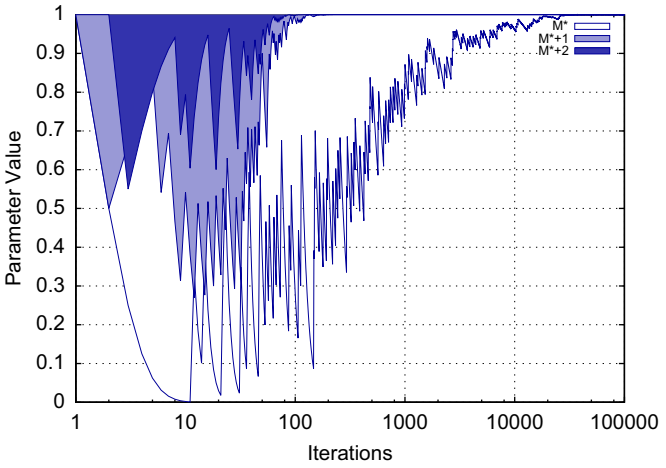

Fig. 4. Evolution of the parameter γ for instance oegl-e4-C.

instances which still have potential of lower bound improvement ($LB_0 \neq UB$). The CPU time was 17.66 min in total and 2.58 s on average, with less than one-fifth of instances above the average. The field *Feas* on Table 3 gives the percentage of feasible solution compared to the number of iterations ran for each instance, thus revealing those instances which have a difficult bin-packing subproblem.

Fig. 4 reveals the evolution of parameter γ for instance oegl-e4-C. The intricacy of seeking for a feasible solution which uses at most M^* vehicles led to the prompt adjustment of γ towards edge demand. As more vehicles are allowed in the solution (M^*+1 and M^*+2), less active is the fitting of γ. It can be noticed that all three curves have a long-term ascendant tendency, which is a consequence of metaparameter g, which is asymptotically driven towards 1 for every feasible solution found (represented by a peak). The rationale of these adjustments is that, as soon as a feasible solution emerge, the heuristic should invest more iterations biasing the edge-selection rule in direction of shortest path cost, thus giving the opportunity for better solutions to arise.

The average probability distribution for the choice of parameter β value is depicted for all three classes of instances (M^*, M^*+1 and M^*+2) in Fig. 5. For the tight M^* class, lower β values were preferred since, as mentioned before, an over responsive ellipse rule may encumber the search for feasible solutions. Still, higher β values are considered more often when feasibility issues no longer concern (M^*+1 and M^*+2).

9. Conclusions

This work introduced OCARP, a NP-hard combinatorial optimization problem of theoretical and practical interest belonging to the family of arc routing problems. At least two applications from literature can be modeled as an OCARP, the Meter Reader Routing Problem and the Cutting Path Determination Problem. The OCARP complexity was proven through a polynomial reduction of the NP-hard CARP.

A reactive path-scanning heuristic was developed for the OCARP. Some of its features are (i) a cost-demand edge-selection rule, self-tuned according with the instance hardness to achieve feasible solutions; (ii) an ellipse rule, which shortens the tours in favor of solution cost; and (iii) a reactive parameter β, responsible for regulating how often the ellipse rule is applied.

In the computational experiments instances from five groups (*ogdb*, *oval*, *oegl*, *oA* and *oB*, based on the CARP instances *gdb* [2], *val* [17], *egl* [36], *A* and *B* [25]), and three classes (according with

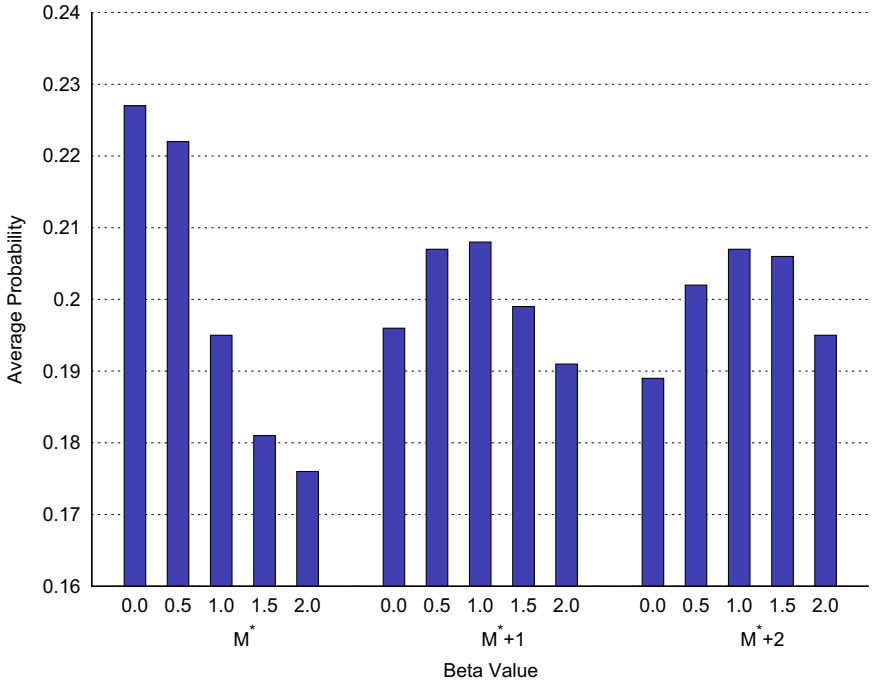

Fig. 5. Probability distribution to choose the value of parameter β.

the number of vehicles available, M^*, M^*+1 and M^*+2, where M^* represents the minimum number of vehicles for a feasible solution), were solved by the proposed path-scanning heuristic, which outperformed two others path-scanning heuristics from literature [20,3] with respect to the overall average deviation from lower bound; optimality was attained for 133 out of 411 instances. In order to improve known trivial lower bounds, a reduced integer OCARP model was solved, deriving better lower bounds for 59 instances.

Future researches should focus on the design of algorithms that can tighten the bounds here introduced, especially the lower bounds. Exact algorithms using column generation and cutting planes approaches should also be investigated.

Acknowledgements

This work was supported by CNPq (Brazilian Research Agency), and IBM (which provided CPLEX 12 license through the IBM Academic Initiative). Special thanks for both anonymous referees for their valuable comments.

References

[1] Golden BL, Wong RT. Capacitated arc routing problems. Networks 1981;11:305–15.
[2] Golden BL, DeArmon JS, Baker EK. Computational experiments with algorithms for a class of routing problems. Computers and Operations Research 1983;10(1):47–59.
[3] Santos L, Coutinho-Rodrigues J, Current JR. An improved heuristic for the capacitated arc routing problem. Computers and Operations Research 2009;36(9):2632–7.
[4] Pearn WL. Augment-insert algorithms for the capacitated arc routing problem. Computers and Operations Research 1991;18:189–98.
[5] Eglese RW, Li LYO. A tabu search based heuristic for arc routing with a capacity constraint and time deadline. In: Osman IH, Kelly JP, editors. Metaheuristics: theory and applications. Kluwer; 1996.
[6] Hertz A, Laporte G, Mittaz M. A tabu search heuristic for the capacitated arc routing problem. Operations Research 2000;48(1):129–35.
[7] Brandão J, Eglese R. A deterministic tabu search algorithm for the capacitated arc routing problem. Computers and Operations Research 2008;35:1112–26.
[8] Lacomme P, Prins C, Ramdane-Chérif W. A genetic algorithm for the capacitated arc routing problem and its extensions. In: Boers EJW, editor. Applications of evolutionary computing. Lecture notes in computer science. Springer; 2001.
[9] Greistorfer P. A tabu scatter search metaheuristic for the arc routing problem. Computers and Industrial Engineering 2003;44:249–66.
[10] Beullens P, Muyldermans L, Cattrysse D, Oudheusden DV. A guided local search heuristic for the capacitated arc routing problem. European Journal of Operational Research 2003;147:629–43.
[11] Hertz A, Mittaz M. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. Transportation Science 2001;35(4):425–34.
[12] Wøhlk S. An approximation algorithm for the capacitated arc routing problem. The Open Operational Research Journal 2008;2:8–12.
[13] Hirabayashi R, Saruwatari Y, Nishida N. Tour construction algorithm for the capacitated arc routing problem. Asia-Pacific Journal of Operational Research 1992;9:155–75.
[14] Longo H, de Aragão MP, Uchoa E. Solving capacitated arc routing problems using a transformation to the cvrp. Computers and Operations Research 2006;33(6):1823–37.
[15] Assad A, Pearn WL, Golden BL. The capacitated chinese postman problem: lower bounds and solvable cases. American Journal of Mathematical and Management Science 1987;7:63–88.
[16] Pearn WL. New lower bounds for the capacitated arc routing problems. Networks 1988;18:181–91.
[17] Benavent E, Campos V, Corberán A, Mota E. The capacitated arc routing problem: lower bounds. Networks 1992;22:669–90.
[18] Belenguer JM, Benavent E. A cutting plane algorithm for the capacitated arc routing problem. Computers and Operations Research 2003;30:705–28.
[19] Wøhlk S. New lower bound for the capacitated arc routing problem. Computers and Operations Research 2006;33(12):3458–72.
[20] Belenguer JM, Benavent E, Lacomme P, Prins C. Lower and upper bounds for the mixed capacitated arc routing problem. Computers and Operations Research 2006;33:3363–83.
[21] Gouveia L, Mourão MC, Pinto LS. Lower bounds for the mixed capacitated arc routing problem. Computers and Operations Research 2010;37(4):692–9.
[22] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, Part II: the rural postman problem. Operations Research 1995;43(3):399–414.
[23] Dror M. Arc routing: theory, solutions and applications. 1st ed. Kluwer Academic Press; 2001. ISBN 0792378989.
[24] Hertz A. Recent trends in arc routing. In: Sharda R, Voß S, Golumbic MC, Hartman IBA, editors. Graph theory, combinatorics and algorithms. Springer US; 2005.
[25] Wøhlk S. A decade of capacitated arc routing. In: Sharda R, Voß S, Golden B, Raghavan S, Wasil E, editors. The vehicle routing problem: latest advances and new challenges, vol. 43. Springer US; 2008. p. 29–48. ISBN 978-0-387-77778-8.
[26] Corberán A, Prins C. Recent results on arc routing problems: an annotated bibliography. Networks 2010;56(1).
[27] Welz SA. Optimal solutions for the capacitated arc routing problem using integer programming. PhD thesis; University of Cincinnati, United States; 1994.

[28] Stern HI, Dror M. Routing electric meter readers. Computers and Operations Research 1979;6:209–23.
[29] Wunderlich J, Collette M, Levy L, Bodin L. Scheduling meter readers for southern california gas company. Interfaces 1992;22:22–30.
[30] Bodin L, Levy L. The arc oriented location routing problem. INFOR 1989;27:74–94.
[31] Bodin L, Levy L. The arc partitioning problem. European Journal of Operational Research 1991;53:393–401.
[32] Moreira LM, Oliveira JF, Gomes AM, Ferreira JS. Heuristics for a dynamic rural postman problem. Computers and Operations Research 2007;34:3281–94.
[33] Pearn WL. Approximate solutions for the capacitated arc routing problem. Computers and Operations Research 1989;16:589–600.
[34] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. John Wiley & Sons; 1990. ISBN 0-471-92420-2.
[35] Prais M, Ribeiro CC. Reactive grasp: an application to a matrix decomposition problem in tdma traffic assignment. INFORMS Journal on Computing 2000;12:164–76.
[36] Li LYO, Eglese RW. An interactive algorithm for vehicle routing for winter-gritting. Journal of the Operational Research Society 1996;47:217–28.